Information Structures for Processing and Retrieving*

Robert A. Colilla and Burnett H. Sams

Radio Corporation of America, Princeton, New Jersey

This paper discusses the problem of conceptualizing an information retrieval system. Broadly characterized, an information retrieval system consists of a file structure to index and hold information, an input language for entering new information into the system or changing the information currently in the system, an interrogating language for couching retrieval requests, a body of programs for performing the various processing tasks, a programming language for specifying new information processing algorithms, and finally a language to control the operations of the system. There is a tremendous variety of possible information retrieval systems. The input and interrogating languages may separately range from binary codes through formal languages to English. The corresponding processing may depend upon specified index terms or may utilize automatic abstracting and indexing techniques. The programming language may range from machine code to an arbitrarily sophisticated compiler code. The control language may be nonexistent, requiring manual loading of programs and data, or may enable self-scheduling of fully automatic parallel processing with dynamic allocation of storage to program and data [1, 2].

To the extent that the input and interrogating language requirements overlap, the input and interrogating languages are able to share terms and syntax. The desirability of incorporating these informational terms and syntax within the programming language to ease the problem of specifying new information processing algorithms is also discussed. Therefore, in the more sophisticated systems, the input, interrogating and programming languages will be incorporated with the control language into a single information processing and retrieval language; in fact, there will be a tendency to incorporate a variety of these languages. (Consider the use of machine code within ALGOL procedures.)

An information retrieval language introduces restraints on the design of data structures and processing programs. We take the point of view that the primary problem is not designing an information retrieval language but rather designing a file structure adequate for the anticipated processing and retrieval on a given class of machines and then designing notations (information processing and retrieval languages) for presenting inputs, requesting outputs, defining algorithms, and controlling operations. These notations may then be made increasingly sophisticated by adding analysis and translation programs to the system. Both approaches require a definition of the information retrieval problem, be it a single problem, a class of problems, or the "generic" processing and retrieval problem. Once the problem has been defined, one may either design a language and trust that a satisfactory implementation can be developed or one may start with the hardware and devise file structures and programming techniques. The latter approach yields a family of systems with increasing capabilities and gives greater assurance that an efficient system can be built. The development of a user's notation is now influenced by the particular file structure and processing patterns chosen, and without detriment to the user a specialized notation may be adopted which will simplify the analysis and translation of the information processing and retrieval language.

We might say, generally, that an information retrieval language for a particular information retrieval system should provide statements for performing all the operations desired to be performed by a user of the system on data described in any way that may be necessary. A general information retrieval language, then, should provide this capability for a large number of different kinds of information retrieval systems. Just how different these systems can be is a measure of the generality of the language.

One might reasonably expect it to be easier to design a language for systems with similar subject matter than with different subject matter since terms and syntax in the first case are likely to be similar. It seems easier for example, to incorporate into one language the statements: "What is the population of New York"? and "What is the temperature of New York at 3 PM on July 10th?" than it is to incorporate "What is the population of New York?" and "What is the best procedure for an American to follow after having an automobile accident in Copenhagen?"

Even for closely related subject matter, however, two systems might be different enough to make different demands on a language that is to be suitable for both systems. Suppose for some system one may ask the question, "How far is it from New York to Philadelphia?" where the answer is to be 90 miles. Suppose another system discriminates among the various routes of travel between New York and Philadelphia and will respond only if a specific route is stated. For example, the system may require a statement of the kind, "How far is it from New York to Philadelphia starting from Times Square, traveling through the Lincoln Tunnel, along the New Jersey turnpike and across the Benjamin Franklin Bridge to Penn Square?" Obviously, if the parameters are changed—e.g. substituting the George Washington Bridge for the Lincoln Tunnel—the answer will be different. The language, therefore, must recognize

* Presented at an Open Technical Meeting on "Design, Implementation and Application of IR-Oriented Languages," held by the ACM Computer Language Committee on Information Retrieval on 20-21 October 1961 in Princeton, N. J.

that route specification is necessary in system two, but should not be present in system one. Its presence in system one gives faulty information since the answer "90 miles" is given regardless of the route expressed. A means for handling inappropriate data, therefore, must likewise be incorporated.

Suppose now that the users of system one decide that it is inadequate for their more detailed investigations and want to incorporate system two while maintaining the old capability, namely, that of recognizing a statement with route unspecified. Suppose, further, they are in agreement that for such a statement—that is, with route unspecified—a tabulation of the ten "most popular" routes is given where "most popular" is statistically determined by frequency of interrogations with route specified. The result is essentially a third system that performs the functions of system two with additional statistical and summarizing operations. The statements acceptable for systems one and two, however, are still acceptable to this third system, but the corresponding functions are now different; hence the interpretation of the same statements for the different systems must be different.

One may conclude from the above examples that if one can design a genera information retrieval language for a number of information retrieval systems, these systems are going to be similar in at least some way other than subject matter. Suppose that instead of trying to design a language for many information retrieval systems, we try to design a general information retrieval system that will be adequate for the solution of a large number of different information retrieval problems. One could then try to design a language for this general system. Each particular system could use as much or as little of the system capability as it chooses. The system should be reasonably efficient regardless of the extent of this use.

We do not claim to have such a system developed but we proceed to describe at least what such a system should be capable of doing to be adequate for a particular class of information retrieval problems that require moderately sophisticated solutions.

First we distinguish between direct and collated document retrieval systems. The object of the former is to get source documents all of which have something in common. Generally, each document is indexed by a set of key terms. The object of the latter is to produce a composite of data extracted and collated from many source documents. Each composite, called a record, is likewise indexed by a set of key terms. For example, if one wanted to know the current status of a particular hurricane, the direct document retrieval system would produce a few dozen weather reports each having something to say about the hurricane. The collated document retrieval system would organize, or would have previously organized, the data in the weather reports and would produce only the current status of the hurricane. If one asked about a tornado that was also discussed on those same weather reports, the direct system would simply present the reports a second time. The collated system would limit itself to the status of the tornado.

Indexing

Both of the above systems require some kind of indexing scheme that can retrieve records (documents or composites) within a short period of time. The simplest kind of indexing associates an index term with a set of records each of which contains the index term. One may wish a more extensive association. For example, if one wants to get information about the consumption of electricity in New Jersey, one would not want to ignore a document concerning the electric consumption in Jersey City. The interrogator knows Jersey City is in New Jersey. He always wants related information about Jersey City or Newark or Trenton when he wants information about New Jersey. The index term "New Jersey", therefore, should subsume the index terms "Jersey City", etc. This particular subsuming is tree-structured and one can devise various techniques for reflecting the desired relationships. Sometimes a tree relationship of subsuming is not adequate. Suppose, in addition to having New Jersey subsume Jersey City, one also wants Greater New York to subsume Jersey City; Jersey City has now become a lattice point in the structure. Dealing with points in a lattice is far more difficult and time-consuming than dealing with points in a tree, so that it is inefficient to view all points as lattice points for the sake of generality. Each problem may have its own particular kind of index term associations which may affect efficiency and it is likely that most problems will have more than one kind. It is concluded, therefore, that the general information retrieval system must have the ability to identify index terms as belonging to a class of index terms where each permits a set of subsuming relations between elements in the class. One might also permit a set of subsuming relations to be defined among the classes themselves.

As soon as one permits classes of index terms, he simultaneously needs a means for identifying the class to which a given term belongs. If the number of classes is small and readily distinguishable, one may choose to require the users of the system to label each term with its class name for each input statement. The internal identification now is simply one of matching on the class name. In some systems the class may be determined by identifying the term. Assuming one can identify classes successfully, one needs to identify terms within a class to see if they are legitimate and to find the subsumed index terms.

If the number of terms in a class is finite, a single glossary will suffice. Each entry in the glossary could indicate which of the remaining entries in the glossary are subsumed index terms. If one anticipates frequent additions to the glossary, it pays to avoid any scheme that depends on the sequence of items in the glossary for these subsumed term indicators.

If, however, the number of items in the class is infinite, one needs an identifying routine. For example, suppose a

system indexes all transactions of $1000 or more by indexing each document under the exact amount of the transaction. Suppose further that for standard requests one subsumes under a given amount all other amounts within a half-dollar of the given amount. That is, for an interrogation using $1065.00 as an index term, all amounts greater than $1064.49 and less than $1065.50 are subsumed.

Although the number of possible index terms in a system may be infinite, the number of documents indexed and the number of index terms which actually reference documents are finite but unbounded. Each information retrieval system needs a list of active index terms with their coupling to the records (documents) they index. Schemes based upon searching the entire record store for each interrogation are excluded from these considerations. If the system were simple, consisting of a single glossary and having only a few subsuming index terms, the glossary itself could serve as the active index list. More generally, this index list is distinct from the glossaries, since some classes having an infinite number of index terms may not require glossaries. By the very fact that index terms may be active or inactive at a particular time, it is necessary to have means for updating the index list. If the list is long, efficiency will dictate that directories and subdirectories be incorporated. The insertion and removal of terms from the active index list and the corresponding maintenance of directories should be entirely automatic. In addition, one may wish to include automatic maintenance of glossaries if additions are frequent.

In cases where the subsumed index terms may form an infinite set, one does not generate them, but rather one examines the index list to see if it contains any index terms that could have been generated. By coding and ordering this index list cleverly, one can substantially shorten this examination. Actually, it may be profitable to introduce more than one index list or even one list for each class of index terms to make this examination efficient.

In general, one does not expect to retrieve on a single index term, but rather on combinations. In fact one might expect to perform arbitrary Boolean functions on sets of index terms. In addition, one may desire some external control over the selection of subsumed index terms for any particular term in some pre-established way. In the example above where one indexed all transactions of amounts larger than $1000, one may want for a particular retrieval only those transactions with amounts greater than $100,000. Each index term, therefore, should permit an operation with parameters that can control the selection of subsumed index terms. The permissible operations must be defined for each index list.

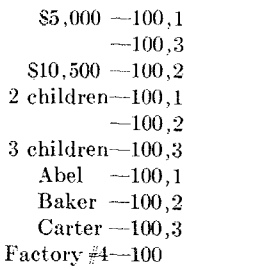

FIG. 1. Example of indexing structure

In many cases, one needs to relate index terms of different classes for retrieval. If one wants to retrieve all employees who earn $10,000 and who have 2 children, it is not sufficient to use the two index terms connected by "and" since one may get employees with two children earning other than $10,000 or employees earning $10,000 with other than two children if more than one person is described in the record indexed.

It is necessary, therefore, to group index classes and to let the terms in the grouped classes index portions of a record, where such portions may be called subjects. For example, consider a document that mentions factory #4 and lists the names of all its employees together with the salary each earns and the number of children each has. Suppose in particular, there are three people Abel, Baker, and Carter who earn $5,000, $10,000 and $5,000 and who have 2, 2, and 3 children respectively. If the document were given the number 100, one could index as illustrated in Figure 1, where the index numbers may consist of one or two parts, the first referencing the document and the second referencing the person in the document. Each person has become a subject within the factory; a subsubject of the subject "Factory". Actually, if more than one factory were mentioned, one would require three parts for each index number, one each for document, factory, and person. Note that the number of persons per factory is unbounded. The grouping of index classes, which parallels the scheme for assigning index numbers paralleling the structure of subjects in documents, is tree-structured.

To summarize indexing, a general information retrieval system should be able to identify terms as belonging to one of several classes of index terms where the classes may be finite or infinite. Further, it must be able to recognize all subsumed index terms that are active where the subsuming relations may be controlled by the interrogator. It must be able to recognize groups of index terms and select only those documents that satisfy the group requirement. It must recognize Boolean functions of groups of index terms and it must maintain the index list, glossaries and directories.

Data Structure

An information store and its interaction with indexing may now be described. The data structure is concerned with the storage of information collected on the various subjects of interest for a particular information processing and retrieval system. The subjects, for example, may exhibit hierarchical relationships such as persons employed by factories or components within assemblies.

The informational units for processing and for retrieval are *fields*. Fields may possess subfields or be combined to form larger fields. A field may exhibit three kinds of variability; the possible values of a field may not all have the same size representation, or more important for present

considerations, a field may require a variable length list of values which are considered either as additional subfields or as alternative values called replications. The smallest fields are called primitive fields. Each primitive field has a specified ordered set of values called *primitive values*. Fields are defined recursively as being either a primitive field or a list of fields; these latter fields are then called subfields. A *field value*, or more briefly *value*, is correspondingly either a primitive value or a list of subfield values. If some subfield is replicated, there is not a uniquely determined value. Accordingly, the *scope of a field*, or more briefly *scope*, is defined as the set of all values where the value ranges over all combinations of subfield replications. The set of all possible field values is ordered in one or more ways as extensions of the orderings defined for each subfield. For example, the field "Personality name" may have the three subfields, "Last name", "First name", and "Middle name", which are further broken down by "Characters" to permit such retrievals as all electrical engineers in Princeton whose first name is "Alan", or alternatively, whose last name ends in the letter "y". Note that a distinction is drawn between the field value "Smith" and the instance of the subject "Person" that is commonly known by the name "Smith". Subjects are organizational entities in correspondence with the objects, individuals, or abstract categories to which the descriptive information pertains. It is convenient to also distinguish—from the fields which describe subjects—the fields, called *auxiliary fields*, which describe the relationship between other fields and the subject. For instance, the population of a city may be accompanied by a date and reference source. In any implemented system there will also be control fields to compensate for the several kinds of format variability allowed.

Some fields are distinguished as the generators of *index classes*. An index class generated by a field consists of the set of all possible field values which are called *index terms* of the index class.

Index class: Manufacturer-Vehicle
Index term: Ford-Truck

The set of index terms is organized into a number of directed linear graphs (not necessarily trees or lattices) by a number of functions on the set of index terms whose values are pairs of sets of ancestor and descendent index terms respectively.

Other fields are distinguished as the generators of *subject classes*. The terms *subject value* and *subject scope* are carried over from the field definitions. An attribute of a subfield which is also a generator of a subject class will be referred to as a *sub-subject attribute*. The members of a subject class are units of storage corresponding to the subject scopes actually retained in the information store of the system. Subjects are the units of indexing. Each index term points to a number of subjects, and each subject is pointed to by at least one, and usually more, index terms.

In the index list, besides identifying a particular subject, it is necessary to identify the class to which the subject belongs; this enables distinguishing between the "Location" in which a person resides and the "Factory" in which he works. The choice of subject classes is matched to the choice of index classes so that the set of subjects to which a pair of index terms jointly apply is precisely the intersection of the two sets of subjects to which the index terms individually apply. Consider for example, the subject class "Automobile ownership", consisting of the two subject classes "Family" and "Auto":

```
Subject—Automobile ownership
Subject—Family
    Name
    Number in family
    Number of drivers
Subject—Auto
    Manufacturer
    Year
```

A possible instance of "Automobile ownership" might be:

```
*Automobile ownership
*Family
    Smith, John Q.
      5
      3
*Auto
    Ford
      62
*Auto
    Volkswagon
      59
```

where the items with asterisks are subject identifiers and the subject "Auto" is replicated.

Note that in a collated document retrieval application it is possible that only a small fraction of the information is deducible from the index-list-subject relationship. In the design of such a system, the statistics of processing and retrieval, the sizes of files, and the nature of the hardware configuration (drum-disc-tape) are prerequisite to making a reasonable determination of indexing structure and the organization of subject storage [3]. In general, the information will be stored in units of records which correspond to subjects (usually with sub-subjects) for which the statistics of record size are well matched to the storage media.

The definitions of field and subject suggest the form of suitable data formats, and correspondingly they suggest the data describing capability of a compiler for the information retrieval system. For many applications the number of fields per record may be rather large, so that one would not want to reserve space in the record for fields which may never have been given values in a particular record. Considering value replications, one cannot know in advance the number of field values to be stored in a given record. When processing considerations impose a maximum record size, the automatic introduction of record continuations and their processing can be made

part of the system to free the user from artificial distinctions. Likewise, the statistics of value sizes may indicate that certain fields should have variable size in order to efficiently utilize machine storage. Each application will require a fresh balancing of storage space versus processing time.

Two elements of information storage, the field and the record, have been introduced. The primitive field is the smallest piece of data that will be named and manipulated by the language. The record is the largest piece of data that will be manipulated as an entity. Records are composed of fields. Within a record, any field or combination of fields may be associated with other fields or combination of fields provided the association is independent of the particular datum that "fills" the field and provided the assocations can be arranged to represent a tree structure for the record. A field is a form that has a value. The value may be any of a set of terms defined by the application or it may be the vacuous symbol. A record is a tree structure of fields independent of the values of the fields. It is to be emphasized that the values for a given field are not required to be tree-structured. It is only the fields themselves that must be so structured in order to provide a scheme for assigning names to data sets stored in the record.

Any piece of information in the system has a name which is factorable into:

1. Record identification
2. Subfield replicate
• Subfield replicate
• Subfield replicate
• Subfield replicate
m. Primitive field replicate

The derivation of names follows directly from the data descriptions of records, subjects and fields. It is recognized that a given term might possess different data descriptions for input, indexing, storage, processing, and output, as demanded for processing and storage efficiency.

Granted the difficulty of constructing a compiler capable of interpreting such data descriptions, processing and retrieval algorithms expressed in terms of these data names could be written in a high-level programming language. The resulting computer programs would generally not be used by themselves as if in a vacuum; rather, a data structure of the complexity suggested above can be justified only by a large information processing system. Such systems can be expected to involve hundreds of thousands of machine instructions and hundreds of millions of characters of data storage. Programs and data will be held in secondary storage and only brought into main memory at the moment of use. In such applications, the processing flow is data-determined and dynamic storage allocation may be necessary [1]. Furthermore, several processing and retrieval requests may be performed in parallel to reduce the cost of each request or to permit the system to serve a number of users on a real-time basis.

Retrieval

When one submits an interrogation saying he wants all information about a particular hurricane, he is saying that he wants to *restrict* himself to documents about that hurricane. Similarly, if he wants information about all factories that have employees who earn $10,000 and who have 2 children and that have employees who earn $5,000 and have 4 children, he is likewise restricting himself to certain records. The terms that do the restricting, we call *restrictors*. Restrictors need not always be index terms. For example, one may not choose to use the date of a document as an index term, but one may still choose to restrict by that date. This assumes that one can find such a date in a given record (document) and test it. This is no problem in a collated system, but it requires some special organizing in a direct document retrieval system.

In collated systems, one may have many restrictors that are not index terms. These terms must be tested after the appropriate record is selected via the index list. This is essentially a two-stage selection procedure. First using the index terms, one gets all records that are associated with the index terms and their subsumed terms. The choice of which restrictors to be used for this purpose and the order in which to use them is not necessarily prespecified, but may be determined for each retrieval by applying heuristics based on previous retrieval efficiencies. The subjects obtained are tested and only those are chosen which satisfy the remaining restrictors. In direct document retrieval systems, one may introduce nonindex term restrictors, but one must supply routines for identifying the items to be tested in the document. One might prefix each document with some header information for this purpose.

Similarly, in a document retrieval system one displays the whole document that one has retrieved and it is up to the interrogator to find what he wants. In a collating system, one may specify those items (the extractors) he wants to see, and it is necessary that the interrogator know what items it is possible for him to see; that is to say, he may extract only those items that can be recognized by the system. This, however, is nothing new, for the same requirement is imposed upon him in using restrictors. The extractors can be expressed in the same manner that the restrictors are expressed. If information is not present, the extractors or groups of extractors may be ignored.

With a little additional effort, one may extract only if the sought data is subsumed by an index term. For example, one may extract a location only if it is subsumed by "New Jersey". This is not quite the same as a roundabout restrictor. A more vivid interrogation that exemplifies this is: "Give me (extract) the names and addresses of (restrict) all persons in factory #4; also give their positions if (conditional extraction) they are among the professional staff". Without the conditional extraction, one could at best obtain two lists, one with the names and addresses of all employees and one with the names, addresses and positions of the professional staff. Of course one can permit Boolean functions of extractors.

PERSONNEL RECORD

Employee Name	Education		Employment		Starting Salary
	School	Degree	Employer	Position	
John Jones	Princeton	BA	Co A	Trainee	$3,000
		MA		Coder	
	Columbia	PhD		Programmer	
	Harvard	LLD	Co B	Programmer	$4,000
			Co C	Senior Programmer	$6,000
				Analyst	
			Co D	IR Specialist	$20,000

FIG. 2. Example of output listing of personnel record

In collating systems, once one has a set of records to be displayed, one has the problem of sorting them in some specified manner. If the field being sorted on appears at most once in every record, there is no problem. Suppose, however, each record from which we have extracted our data has as subject a particular apartment house in some city and suppose that among the data in the record are the names of all the tenants and the address of the apartment house. If the interrogation reads "In alphabetical order, list the names and addresses of all people who live in apartment houses in that city", artificial records are created for sorting, duplicating the address of the house for each person who lives in it. Further complications arise if more than one level of sorting is specified at one time.

Display

Display includes preparation of extracted data for tabular presentation using line printing devices. The retrieved information is extracted as subsets of selected subjects. This information therefore has subject structure and must be printed so as to reflect that structure. That is, replications of a particular sub-subject or field are vertically aligned and are vertically spaced so as not to conflict with each other. This also applies to values and repeated fields that exceed alotted column width. Figure 2 shows a personnel record for John Jones. Column 1 consists of the field "Name" that comprises the subject "Employee". Columns 2 and 3 show the fields "School" and "Degree" of the subject "Education". The remaining columns 4–6 show the fields "Employer", "Position", and "Starting salary" for the subject "Employment". No coordination is made among the three subjects "Employee", "Education" and "Employment" within a single record. The subject "Education" permits replications on the field "Degree". These replications are coordinated with the value of school name. Similarly, "Employment" permits replications on the field "Position". These replications are coordinated with the values of "Employer" and "Starting salary".

Summary

In a data organization for an information processing and retrieval system, the recurring problems of structure and scope pervade all aspects of the processing. This is necessary if the system is to permit the variety of semantic relationships required to effect moderately sophisticated solutions for a class of information retrieval problems. These considerations have suggested a flexibility and variability not yet available in any system or computer language. The problems of constructing such programs are considerable, and while such programs are not at hand, they are on the horizon.

REFERENCES

1. SAMS, B. H. Dynamic storage allocation for an information retrieval system. *Comm. ACM 4* (Oct. 1961), 431–435.
2. CHEATHAM, T. E., JR.; COLLINS, G. O., JR.; AND LEONARD, G. F. CL-1, an environment for a compiler. *Comm. ACM 4* (Jan. 1961), 23–28.
3. MINKER, J. M. Implementation of large information retrieval problems. Gordon Research Conference, New Hampton School, New Hampton, N. H. (July, 1961).

An Information System With The Ability To Extract Intelligence From Data*

T. L. Wang

General Electric Company, Syracuse, New York

Introduction

An information system with the ability to extract intelligence or knowledge from a large mass of data has been developed and instituted. This information system was developed under the auspices of the Minuteman High Reliability Component Program, the objective of which is to improve the reliability of the components used in the Minuteman System by two to three orders of magnitude.

To meet this challenge, it is necessary to collect and analyze eventually about 200 million digits of variable data concerning a specific type of transistor alone. This large mass of data, once collected, should contain almost all the answers that evaluation engineers and statisticians must know in order to improve the reliability of this particular transistor.

* Presented at an Open Technical Meeting on "Design, Implementation and Application of IR-Oriented Languages," held by the ACM Computer Language Committee on Information Retrieval on 20–21 October 1961 in Princeton, N. J.